NASA Contractor Report 187528
ICASE Report No. 91-19

ICASE

DOMAIN DECOMPOSITION WITH LOCAL MESH REFINEMENT

William D. Gropp
David E. Keyes

Contract No. NAS1-18605
February 1991

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, Virginia 23665-5225

Operated by the Universities Space Research Association

National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665-5225

DOMAIN DECOMPOSITION WITH LOCAL MESH REFINEMENT[1]

William D. Gropp[2]
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439
and
David E. Keyes[3]
Department of Mechanical Engineering
Yale University
New Haven, CT 06520

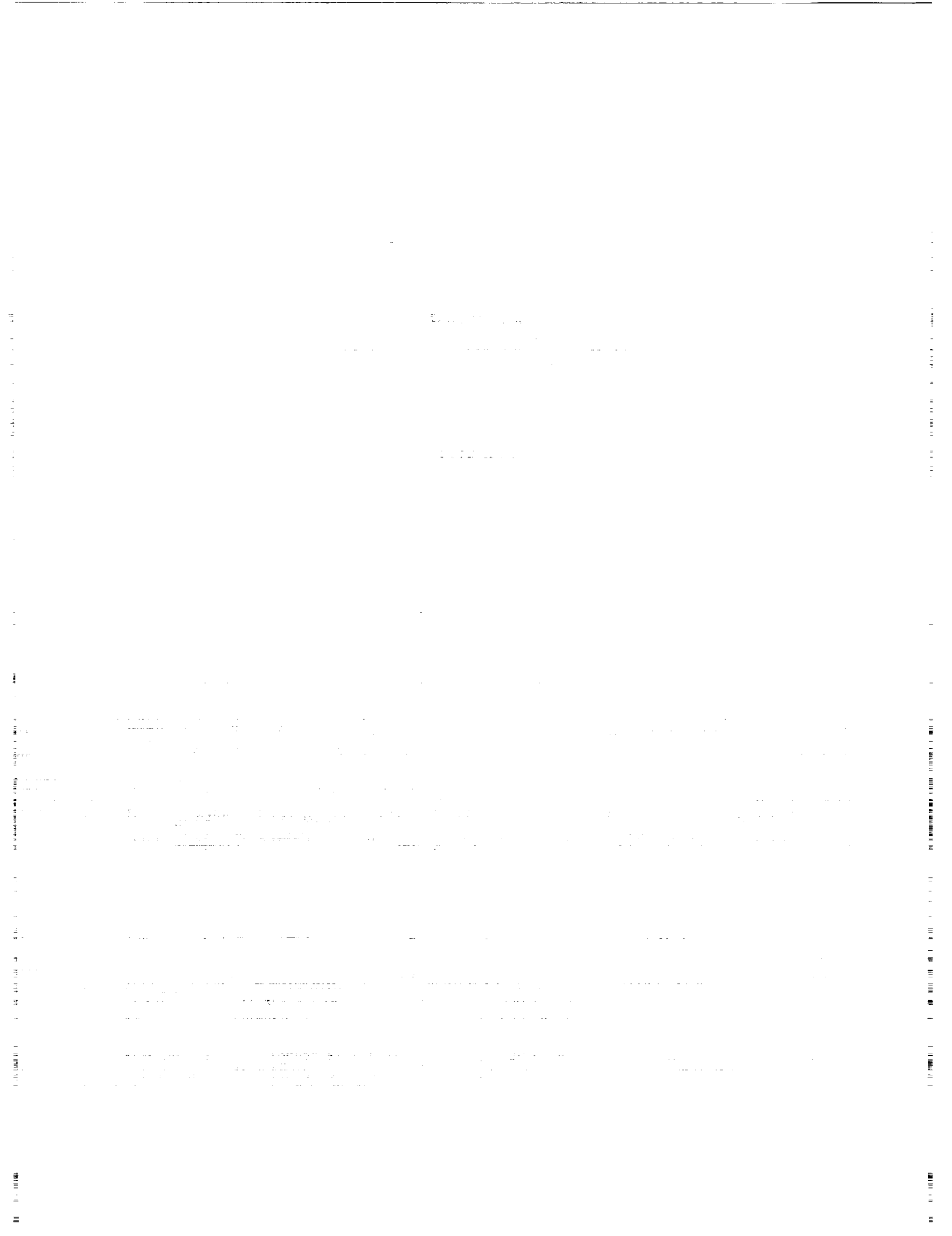
ABSTRACT

We describe a preconditioned Krylov iterative algorithm based on domain decomposition for linear systems arising from implicit finite-difference or finite-element discretizations of partial differential equation problems requiring local mesh refinement. To keep data structures as simple as possible for parallel computing applications, we define the fundamental computational unit in the algorithm as a subregion of the domain spanned by a locally uniform tensor-product grid, called a tile. In the tile-based domain decomposition approach, two levels of discretization are considered at each point of the domain: a global coarse grid defined by tile vertices only, and a local fine grid where the degree of resolution can vary from tile to tile. One global level and one local level provide the flexibility required to adaptively discretize a diverse collection of problems on irregular regions and solve them at convergence rates that deteriorate only logarithmically in the finest mesh parameter, with the coarse tessellation held fixed. A logarithmic departure from optimality seems to be a reasonable compromise for the simplicity of the composite grid data structure and concomitant regular data exchange patterns in a multiprocessor environment. We report some experiments with up to 1024 tiles, comment on the evolution of the algorithm, and contrast it with optimal nonrefining two-level algorithms and optimal refining multilevel algorithms. Computational comparisons with some other popular methods are presented.

[1]Major revision of Yale University Department of Computer Science Research Report 726, August 1989 version.

[2]The work of this author was supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under Contract W-31-109-Eng-38, by the Office of Naval Research under Contract N00014-86-K-0310, and by the National Science Foundation under Contract DCR 8521451.

[3]The work of this author was supported in part by the National Science Foundation under Contracts EET-8707109 and ECS-8957475 and by the National Aeronautics and Space Administration under NASA Contract NAS1-18605 while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23665.



1. Introduction

The combination of domain decomposition with preconditioned iterative methods provides a framework that extends the usefulness of numerical techniques for certain special partial differential equation (PDE) problems to those of more general structure. Nonsmooth features, nonseparable geometries, or massive sizes of practical problems limit the application of many "standard" numerical techniques. Direct methods are rapidly defeated by problem size. "Fast" methods which take advantage of special coefficient and grid structure often do not apply globally. Iterative methods often depend for efficient implementation on regular grids that, if global in extent, are inconsistent with accurate and economical resolution of the physics of the problem. However, the domains of problems with these features can often be decomposed into smaller subdomains of simpler structure, increasing the utility of extant software libraries, particularly as components of preconditioners. Moreover, the domain decomposition can be made to produce a convenient mapping of many problems onto medium-scale parallel computers. Our primary focus in this paper is the incorporation of spatially varying mesh refinement requirements into a domain decomposition algorithm based on finite differences. We illustrate the convergence behavior of the algorithm on a variety of two-dimensional elliptic PDE problems,

$$\mathcal{L}u = f \text{ on } \Omega, \text{ with } au + bu_n = g \text{ on } \mathrm{d}\Omega, \tag{1.1}$$

including non-selfadjoint, nonseparable geometry cases. We also point out features of the method that are relevant to a parallel implementation. We defer most discussion that would be distinctly architecture- and performance-related to a companion paper [25].

Many PDE problems that are "large" in the discrete sense are so because the continuous problems from which they are generated require resolution of several different length scales for the production of a meaningful solution. The value of compromising between the extremes of globally uniform refinement (which leads to simple and usually vectorizable algorithms but wastes time and memory) and pointwise adaptive refinement (which minimizes the discrete problem size for a given accuracy requirement but leads to complicated data structures) has been recognized for some time and described in contexts too numerous to acknowledge fairly. Locally Uniform Mesh Refinement (LUMR) characterizes one such class of discretizations, based on composites of highly structured subgrids. Many treatments of LUMR in the literature pertain to explicit methods for transient problems, a class with its own advantages (see [3] and references therein) and limitations [44] which is somewhat distinct from ours. Implicit treatments of locally regular refinement for elliptic problems include approaches arising out of classical multigrid ([7]; see [36] for a concise state-of-the-art treatment), a nonconforming spectral technique [34], and methods rooted in iterative substructuring for finite element problems [5, 15].

Computationally practical locally uniform grids are usually expressible as the union of a coarse uniform tensor-product grid covering the entire domain with one or more refined tensor-product grids defined over subregions, including the possibility of multiple, nested levels. Generalizations of these grids within the LUMR framework include allowing the grids at any particular level of refinement to themselves be the union of tensor-product subgrids, and reinterpreting "uniform" as "quasi-uniform" to allow general curvilinear coordinates for custom body- or solution-fitting. Few *parallel* implementations of schemes of this generality have been reported thus far. Selected for consideration here is a structurally restricted form of LUMR in which refinement occurs exclusively within complete cells of a quasi-uniform coarse grid, as described in Section 3 below.

The goal of the present contribution is an LUMR methodology with starkly simple data structures for efficient portability to a variety of parallel machines. Current implementations on two distributed- and two shared-memory parallel machines share approximately 98% common code measured by line count, inclusive of comments, exclusive of standard libraries. The methodology

borrows from the mesh refinement and domain decomposition literature and from the authors' own experience in these areas and in parallel computation [22, 23, 32]. The serial arithmetic complexity bows somewhat to modularity, portability, and overall parallel performance, in which we include both efficiency and total execution time. For example, by refining only in units of full coarse-grid cells, we often impose a tendency towards refinement in subregions where it would be unnecessary from the viewpoint of truncation error alone. As another example, the convergence rate of many domain decomposition algorithms is mildly dependent upon a coarse-grid resolution which may be chosen with criteria beyond convergence rate, such as the balance of work among multiple processors. Our algorithm therefore does not scale (with constant problem size) to indefinitely large numbers of processors, but it does sit comfortably on today's MIMD supercomputers. We comment in the final section about a hybridized two-level algorithm suitable for massive parallelism on an MIMD cluster of SIMD subclusters.

Prior to the discussion of LUMR, Section 2 describes a domain decomposition algorithm employing "nearly" parallel preconditioners in conjunction with generalized minimal residual (GMRES) iteration, a nonstationary method not dependent upon operator symmetry. In two dimensions, the preconditioner involves three phases: a global coarse grid solve, independent solves along interfaces between subdomains, and independent solves in the subdomain interiors. The global coarse-grid solve is an essential feature, as it provides the only global exchange of information in the preconditioner itself. We introduce a simple "tangential" operator preconditioning for the subdomain interfaces that is preferable to the interface probe preconditioning advocated in our earlier work on convective-diffusive systems with stripwise decompositions [31]. We also prefer exact subdomain solves to incomplete factorizations. These "exact solves" can be performed by multigrid if the subdomains become too large for direct methods. For multicomponent problems in which source terms codominate with convection and diffusion, block incomplete factorization may also be an economical subdomain solver.

The main body of the paper (Section 4) is the collection of numerical experiments on two-dimensional elliptic boundary value problems (BVPs). The experiments include reentrant domains, non-selfadjoint operators, and mixed boundary conditions. Up to 1,024 coarse-grid elements, called *tiles*, are used. The last two subsections of Section 4 compare the boxwise decompositions used throughout the paper with stripwise decompositions exploiting physical anisotropies, as well as with some conventional undecomposed solvers.

Section 5 indicates some future directions for this methodology.

2. An Iterative Domain Decomposition Algorithm

Preconditioned iterative methods and domain decomposition provide a framework that includes a wide class of algorithms. This framework comprises four elements:

1. a global operator arising from the discretization of the PDE (or system of PDEs);
2. an approximate inverse, or preconditioner, for the global operator;
3. an iterative method requiring only the application of the preconditioned operator; and
4. a geometry-based partition of the discrete unknowns so that size, locality, and uniformity can be exploited in forming the action of the preconditioned operator.

Since the numerical analysis literature contains many successful discretization schemes and iterative methods specialized for different operator properties, such as the presence or absence of definiteness and symmetry, the recent burgeoning effort in iterative domain decomposition algorithms has concentrated primarily (though not exclusively) on the interaction of the second and fourth of these elements. In the parallel context, this is a natural preoccupation because the bottleneck to parallelism usually (though not exclusively) lies in the requirement of the global transport of information in the preconditioner.

Subsection 2.1 establishes block matrix notation corresponding to the decomposition of the domain. The rest of Section 2 supplies detail important to specialists and to implementors of the algorithm, but can be passed over in a casual reading.

2.1. Iterative Methods and Operator Structure

Many of the numerical examples described in Section 4 rule out the use of iterative methods based on symmetry but permit the assumptions of definiteness and diagonal dominance. In particular, full or incomplete factorizations of preconditioner matrix blocks can be undertaken without pivoting. Because of its robustness, we adopt the parameter-free generalized minimal residual (GMRES) method [42] as the outer iteration. The main disadvantages of GMRES, its linear and quadratic (in iteration index) memory and execution time requirements, respectively, must be mitigated by scaling and preconditioning. For other acceleration schemes, such as Chebyshev, the memory and execution time requirements may be only constant and linear, respectively; however, GMRES dispenses with the difficulty of estimating parameters. The recently proposed, parameter-free, bounded recursion Bi-CGSTAB method [45] combines the above-mentioned advantages and deserves further study. In preliminary tests we have found it usually to be competitive in execution time with GMRES, but it can in some instances be substantially slower. Other methods for the iterative solution of nonsymmetric systems, such as QMR [20], also deserve broader investigation. In solving $Au = b$ each of these methods requires an initial iterate for x that it improves through repeated calls to a routine forming the product of A with a direction vector. For improved convergence we employ a change of variables, iteratively solving $(AB^{-1})y = b$ for y and then $Bx = y$ for x. Here, B is a right preconditioner matrix, whose inverse action should be convenient to compute and should cluster the eigenvalues of AB^{-1}. A arises from an FD, FE, or FV discretization, with a local stencil. The stencil is regarded as uniform in this section and generalized in Section 3.

The type of domain decomposition used here involves unit aspect ratio subdomains, as opposed to thin strips joining opposite sides of a domain; therefore, interior subdomain vertices are created. We denote all subdomain vertices "cross points" but distinguish between interior and boundary cross points. Ordering the interior points as well as the physical boundary points other than cross points first, the interfaces connecting the cross points plus the cross points on the boundary next, and the interior cross points last imposes the following outer tripartition on the global discrete operator A:

$$A \equiv \begin{pmatrix} A_I & A_{IB} & A_{IC} \\ A_{BI} & A_B & A_{BC} \\ A_{CI} & A_{CB} & A_C \end{pmatrix}. \tag{2.1}$$

Note that the partitions vary greatly in size. If H is a quasi-uniform subdomain diameter and h a quasi-uniform fine mesh width, the discrete dimensions of A_I, A_B, and A_C are $\mathcal{O}(h^{-2})$, $\mathcal{O}(H^{-1}h^{-1})$, and $\mathcal{O}(H^{-2})$, respectively. The numerical experiments described below employ a five-point stencil, (extended in [33] to second-order upwind differencing with a skew six- or seven-point stencil). No cross-derivative terms appear; therefore, there are no corner points in the stencil, and blocks A_{IC} and A_{CI} may be set to zero.

The outer structure of our preconditioner B may be a conformally partitioned block upper triangular matrix:

$$B = \begin{pmatrix} \tilde{A}_I & \tilde{A}_{IB} & \tilde{A}_{IC} \\ 0 & B_B & \tilde{A}_{BC} \\ 0 & 0 & B_C \end{pmatrix}, \tag{2.2}$$

whose components are elucidated in the next subsection. The application of B^{-1} to a vector $v = (v_I, v_B, v_C)^T$ consists of solving $Bw = v$ for $w = (w_I, w_B, w_C)^T$. It begins with a cross-point solve with B_C for w_C. This updates through $\tilde{A}_{BC}$ the right-hand sides of a set of independent

interface solves for subvectors of w_B. These, in turn, update the right-hand sides through $\tilde{A}_{IB}$ of a set of independent interior solves for subvectors of w_I. For a stencil with corner points, $\tilde{A}_{IC}$ would be nonzero, and the cross-point result would also update the interior block right-hand sides. Within the preconditioner, however, there is no dependence of the interface solution upon the result of the interior solution, or of the cross-point solution upon either. This fact distinguishes the method from [6] and [9] and means that the $\mathcal{O}(h^{-2})$-sized block of the preconditioner is visited only once per iteration.

A useful optimization is available when the tilde quantities in the top row are taken equal to their tilde-free counterparts. In this case, it is readily verified that the right-preconditioned form of the operator is

$$AB^{-1} = \begin{pmatrix} I & 0 & 0 \\ A_{BI}A_I^{-1} & (A_B - A_{BI}A_I^{-1}A_{IB})B_B^{-1} & (I - (A_B - A_{BI}A_I^{-1}A_{IB})B_B^{-1})\tilde{A}_{BC}B_C^{-1} \\ 0 & A_{CB}B_B^{-1} & (A_C - A_{CB}B_B^{-1}\tilde{A}_{BC})B_C^{-1} \end{pmatrix}. \quad (2.3)$$

The identity block row means that $\mathcal{O}(h^{-2})$ of the unknowns in the Krylov vectors can go untouched (except for scaling) throughout the entire solution process until the preconditioning is unwound in the final step, after the interface and cross-point values have converged. Since A_I^{-1} is needed to advance the solution on these separator sets, we cannot escape solving subdomain problems, but substantial arithmetic work can be saved.

2.2. Components of the Preconditioner

The derivation of the coefficients of the preconditioner blocks is as follows. The cross-point operator B_C is simply an H-scale coarse-grid discretization of the continuous PDE. To accommodate Neumann or Robin boundary conditions, we include physical boundary points lying at subdomain vertices in the cross-point system in this step. (Later the boundary cross-point values are overwritten with the results of more accurate h-scale data from the interface solve. This distinction is, of course, moot for Dirichlet data but important for boundary conditions involving spatial gradients.) The cross-point system right-hand side at interior points can be taken as the injected vertex values of the fine-grid right-hand side, though we remark on a better choice below. The current implementation supports a direct solve on the coarse-grid system. If strip decompositions are used, there is no cross-point system, and the lower-right block of the preconditioner is simply the interface system described next.

Unlike the coarse-grid and subdomain interior problems, which possess the full physical dimension of the domain of the underlying PDE and thus inherit a literature full of preconditionings, the lower-dimensional interfacial equations are properly derived from a pseudo-differential trace operator. A theoretically well developed approach for preconditioning the interfaces has been developed for the non-selfadjoint case in a setting requiring two sets of subdomain solves per iteration [9], but we have experimented instead with three approaches referred to as tangential, truncated, and interface probe. The tangential interface preconditioner is the one-dimensional discretization of the terms of the underlying operator that remain when the derivatives normal to the interface are set to zero. It is equivalent to solving a two-point BVP with boundary conditions inherited from the interior cross-point values or from the physical boundary. The truncated interface preconditioner is a discretization of the full underlying operator, with the coefficients associated with noninterfacial unknowns set to zero. The interface probe preconditioner has been described elsewhere [10, 12, 31] as a low-bandwidth approximation to the true capacitance matrix of the interfacial unknowns in the ambient matrix corresponding to the degrees of freedom of the interface itself and the two subdomain interiors on either side. For all of the results in this paper, the tangential preconditioner is used because it performed better than the others on average in our tests, as described in more detail in [24].

The subdomain interior equations consist of approximate fine-grid discretizations of the PDE over local regions, with physically appropriate boundary conditions along any true boundary segments and Dirichlet boundary conditions derived from the already-available w_B at artificial interfaces. Only first-order interior differences are accommodated in the physical boundary conditions of the preconditioner, though first- *or* second-order boundary conditions may be elected in the operator A. The current implementation supports full LU Gaussian elimination with both banded and sparse data structures, fast cyclic reduction, incomplete LU decomposition, and modified incomplete LU decomposition. To maintain a reasonable scope, we concentrate on full elimination results here. Full elimination on the interiors yields the best iteration counts, though not always the best execution times (for large H/h). Like the interface solves, each subdomain interior solve may be performed independently.

2.2.1. A Better Variant of the Cross-Point System

In keeping with an exposition that is as independent as possible of particular discretization techniques, the right-hand side of the cross-point system was assumed above to be the injected vertex values of the fine grid weighted by the subdomain areas instead of the grid-cell areas. It is necessary, however, to rely on a finite element discretization with a hierarchical basis to properly motivate the construction of a better cross-point system. In particular, we have obtained faster convergence by using the function space decomposition approach of [6], which yields essentially the same coefficient block B_C but replaces the simple injection of fine-grid values with ramp-weighted averages of interface values along all interfaces feeding a given cross point. Specifically, the element of the right-hand side v_C corresponding to an interior subdomain vertex is a discrete approximation to one-quarter of the sum of four line integrals of the form

$$\frac{2}{H}\int_0^H v(s)\left(1-\frac{s}{H}\right)\,ds\,, \tag{2.4}$$

where s parameterizes the interfaces leading from the vertex in question. This leads to the following sequence of steps to produce a preconditioned matrix vector product u from input v, where $v = (v_I, v_B, v_C)^T$:

First, v_C is reweighted according to (2.4). The reweighting has the matrix representation $v' = Cv$, where

$$C = \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & K & J \end{pmatrix},$$

J is diagonal with all positive elements, all elements of K are nonnegative, and the row sums of K and J together give unity. Then, as above, we solve for $w = B^{-1}v'$ and multiply by A to get $u = Aw$. Thus, the preconditioned matrix vector product is $u = AB^{-1}Cv$. Treating everything apart from A itself as the effective preconditioner Q, we find that $Q^{-1} = B^{-1}C$, or equivalently, $Q = C^{-1}B$. C^{-1} is straightforwardly seen to be

$$C^{-1} = \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & -J^{-1}K & J^{-1} \end{pmatrix},$$

so that

$$Q = \begin{pmatrix} \tilde{A}_I & \tilde{A}_{IB} & \tilde{A}_{IC} \\ 0 & B_B & \tilde{A}_{BC} \\ 0 & -J^{-1}KB_B & J^{-1}(B_C - K\tilde{A}_{BC}) \end{pmatrix}.$$

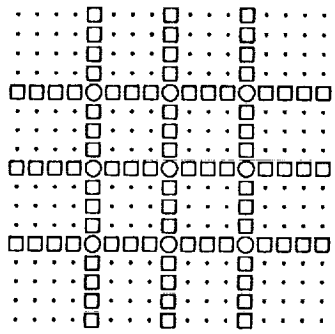


Figure 1: A sample partition of the unknown vector u into u_C (circles), u_B (squares), and u_I (dots) for a 4×4 decomposition of a square into tiles.

When tilde-free quantities are used in the first block row, AQ^{-1} has a first block row equal to the identity. Thus, the remark following equation (2.3) about not touching the upper portion of the Krylov vectors, except for scaling, remains valid. Though the preconditioner with ramp weighting of the right-hand side of the cross-point system is no longer strictly block triangular, it still requires only one solve with A_I per iteration.

2.2.2. A Simple Example of an (A, B) Pair

An example may be the most effective way to indicate how AB^{-1} is applied to a vector, this being the central action in any preconditioned iterative method for a finite local discretization of (1.1). For simplicity, consider $\mathcal{L}$ as the negative of the Laplacian, $b = g = 0$, and Ω as the unit square partitioned uniformly into 4×4 subdomains of 4×4 cells each (i.e., $h = 1/16$ and $H = 1/4$), and employ second-order finite differences. (We emphasize that for selfadjoint and/or constant coefficient problems such as this, several better techniques exist than the one now illustrated.)

As shown in Figure 1, there are 17^2 degrees of freedom, including 3^2 interior crosspoints in u_C, 24×3 interior interface points in u_B, and 16×3^2 regular interior points in u_I. Of the remaining 64 boundary points, the 12 attached to interfaces are grouped with u_B, and the balance with u_I. The matrix A is a simple permutation of pent$\{-1 \ldots -1\ 4\ -1 \ldots -1\}$, except that boundary rows are replaced with corresponding identity rows. The right-hand side is the corresponding permutation of $h^2 f_{ij}$, again with boundary conditions imposed in the appropriate rows. The block B_C is pent$\{-1 \ldots -1\ 4\ -1 \ldots -1\}$. B_B is block diagonal with 24 copies of tri$\{-1\ 2\ -1\}$, some augmented with boundary identity rows. $\tilde{A}_{BC}$ is the corresponding block of A, and the entire interior row set is that of A. Before B^{-1} is applied to a vector v, the components v_C are replaced with the interface averages described above and scaled by H^2. Following the block triangular backsolve with (2.2), a simple matrix-vector multiply is done with (2.1).

2.3. Parallelism in the Preconditioner

We note that the permutation into block matrix form described in this section is a purely formal one for notational convenience. The data structure used in a computer implementation is a local natural ordering of gridpoints within a natural ordering of tiles, as detailed in Section 3 below. The parallelism within $\tilde{A}_I$ and B_B is not visible at the level of blocking in (2.2), but the parallel bottleneck represented by communication-intensive B_C and the sequential use of that result in multiplications with $\tilde{A}_{BC}$ (and, generally, $\tilde{A}_{IC}$) blocks *is* evident. We mention variants of the algorithm that alleviate this bottleneck at the price of some extra local work and extra storage.

The solve with B_C itself can be performed in any of three ways: redundantly on each processor after broadcasting the required coefficient data, with single-threaded code between collecting

the coefficients on a single processor and redistributing the results, or in a fully (or partially) distributed fashion. Determination of the most efficient technique is generally decomposition- and network-dependent, since problem size and computation-to-communication ratios enter the complexity estimate in nonisolable ways. Some global data exchange is necessary in this phase, so it may be desirable to prevent idling on a given multiprocessor to allow the remaining local exchange phases to proceed before the cross-point results are available.

The sequentiality of the cross-point solve can be broken by the following technique, which exploits the relatively small size of the cross-point system. Lumping the balance of the unknowns together, let (2.2) be condensed to

$$B = \begin{pmatrix} B_h & B_{hH} \\ 0 & B_H \end{pmatrix},$$

where B_h contains the upper 2×2 blocks of (2.2), and B_H is just another name for B_C. Consider the application of the preconditioner

$$B \begin{pmatrix} w_h \\ w_H \end{pmatrix} = \begin{pmatrix} v_h \\ v_H \end{pmatrix},$$

with the sequential solution

$$w_H = B_H^{-1} v_H,$$
$$w_h = B_h^{-1}(v_h - B_{hH} w_H).$$

A preprocessing step can compute and store the vectors $g_k = (B_h^{-1})(B_{hH})e_k$, $k = 1, \ldots, K$, where there are K interior cross points and e_k is the k^{th} unit vector in this K-dimensional space. Once $w_H = B_H^{-1} v_H$ and $w_h^{(1)} \equiv B_h^{-1} v_h$ are independently solved for, we can (through local computations) form $w_h = w_h^{(1)} - \sum_k (w_H)_k g_k$. By construction, the support of each g_k is limited to the four tiles sharing vertex k, and the cross points possess a four-coloring that allows the g_k to be computed in just four sets of independent subdomain solves (for a scalar PDE). This process was inspired by, and has an interpretation in terms of, function space decompositions. Indeed, the function space framework is critical in generalizations to multilevel preconditioners, but for a two-level preconditioner the algebraic description above is sufficient.

3. Mesh Refinement by Tiles

This section describes a simple mesh refinement philosophy based on a regular tessellation of two-dimensional domains into subdomain "tiles." A tile is a tensor-product of half-open intervals in each coordinate direction, except that a tile abutting a physical boundary along what would ordinarily be one of its open edges is closed along that edge. Each tile possesses its own interior, at least two of its four sides, and at least one of its four corners and is locally discretized on a tensor-product grid. Although the specific convention is arbitrary, we assume for definiteness that in its own local right-handed coordinate system, each tile contains its origin and its x and y axes (see Figure 2).

We require that the cross points be embeddable in a tensor-product global quasi-uniform coarse grid, from which only points lying exterior to the (possibly multiply-connected) boundary are missing. Irregular tiling patterns such as in Figure 3b are ruled out for convenience in setting up the coarse grid system and keeping the code that manages the interfacial data exchanges short. However, there is no requirement that the domain itself be of tensor-product type; the decomposition in Figure 3a is permissible. Without coordinate stretching and other body-fitted coordinate transformations, the embedding requirement would generally enslave the granularity of the decomposition to the geometric complexity of the domain, a situation that we wish to avoid since

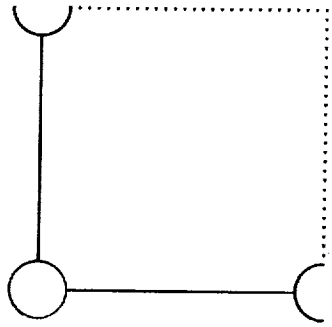
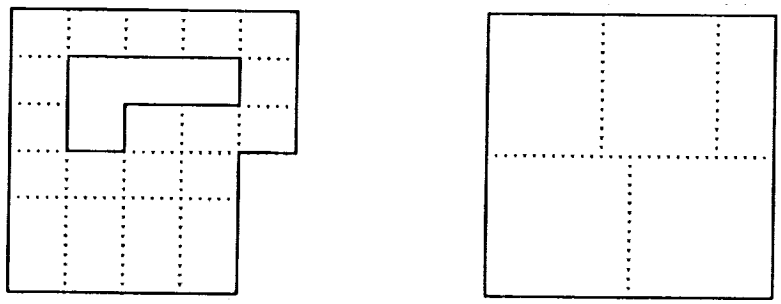


Figure 2: The anatomy of a tile. Unless closed by a physical boundary, a tile is open along its high-x and high-y perimeter.



(a) (b)

Figure 3: Sample tessellations: (a) is permissible, (b) is not.

granularity has important implications on load balance and convergence rate. Though we have yet to fully implement them, domain-wide coordinate transformations represent a simple extension in principle. From an algebraic point of view, an orthogonal body-fitted coordinate transformation is indistiguishable from a perturbation to the operator coefficients. Preserving orthogonality should create less of a strain on a mesh generator acting over local regions than it does in much current practice using global mappings.

Associated with each tile is the data defined over a quasi-uniform grid covering its portion of the domain and a set of operators for executing its block-row portions of the preconditioner solve, as described in Section 2. In our object-oriented approach, these operators can vary widely from tile to tile. In our present examples, however, we assume that the grids covering individual tiles share a common parent uniform tile (of arbitrary discrete size) and are refined only in powers of 2. We can therefore later indicate refinement levels using the graphical shorthand of Figure 12 where the integer indicates the logarithm of the refinement ratio.

3.1. Tile-Tile Interfaces

To minimize restrictions on the structure of adjacent tiles (and to eliminate redundant communication between tiles in a multiprocessor implementation in which different tiles will generally be assigned to different processors), each tile stores and maintains, in addition to its own data, the data associated with a buffer region of phantom points equal in width to one-half of that of its associated discrete stencil. Figure 4 illustrates the buffer unknowns for a five-point stencil, superimposed on Figure 2. With the exception of these redundant phantom points, each point of the domain is uniquely associated with a single tile.

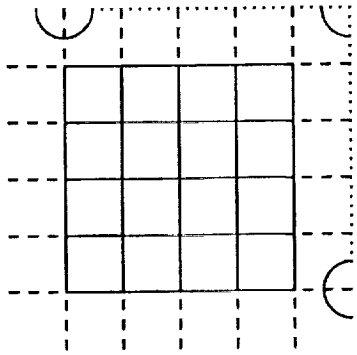


Figure 4: Sample tile, showing the computational buffer region (dashed extensions) required for the completion of standard five-point stencils centered at the points of the local grid.

Data at the phantom points is supplied in a manner dependent upon the internal structure and refinement ratios of the associated adjacent tiles. A finer tile obtains biquadratically interpolated data from its coarser neighbor. Since the problems studied here involve second-order operators, this allows the use of conventional finite-difference techniques in generating the difference equations at the subdomain interfaces. (Bilinear interpolation alone would limit the potential accuracy of a second-order differencing scheme, as observed in some preliminary experiments.) A coarser tile obtains its data by simple (unweighted) injection. That is, the value at the point in the finer neighboring tile that lies on the extended coarser tile stencil is scaled appropriately and used in the coarser grid.

We note that such a simple scheme neither guarantees discrete flux conservation nor delivers a symmetric A for a selfadjoint $\mathcal{L}$. However, the algebraic method does not depend on either property. The focus of this paper is on the solution of a consistent set of discrete equations. More careful attention to the conservation properties of the discretization has been given in the context of locally regular refinement in [18] and [36], for instance.

Each iteration of GMRES requires multiplying with A, which involves at most nearest-neighbor data exchanges between tiles to complete the local stencils, and solving with B, which likewise requires only nearest-neighbor data exchanges to form right-hand sides, *apart from* the globally cooperative task of solving with B_C.

The selection of refinement criteria is a much-studied, yet still open problem; see [29] and the collections [1, 19] for representative work in this area. The refinement criteria, however, are orthogonal to the equation-solving aspect considered here, except to the extent that a part of the computational work required by one of these tasks may be a by-product of the other. In the examples, "good" refinement strategies can be done manually.

In general, tile interfaces can be the site of changes in the discretization besides just the refinement level. For instance, the discrete stencil can change order at interfaces. Even the form of the operators or the number can change at interfaces while still preserving the subdomain uniformity required for efficient subdomain solution algorithms. As a motivational example, a reacting flow problem frequently consists of large regions in which there is only transport of mass, momentum, and thermal energy but no reaction among stable constituents to all adequate orders of approximation. In other regions it is essential to retain a full set of composition variables, including trace radicals, and reaction terms must also be retained in the equations. To accommodate such generality, the routines that pack the buffer regions are responsible for providing the necessary mappings.

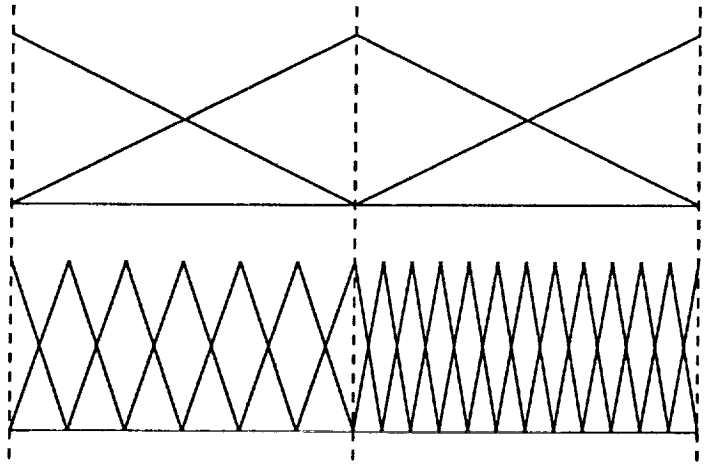


Figure 5: One-dimensional schematic of the tile basis functions.

3.2. Physical Boundaries

For generality, the equations for the physical boundaries are incorporated into the overall system matrix, including Dirichlet conditions. Our implementation allows inhomogeneous Robin boundary conditions at all boundary points, namely,

$$a(x,y)u + b(x,y)\frac{\partial u}{\partial n} = c(x,y).$$

Either first- or second-order one-sided difference approximations to the normal derivative term may be employed in the actual operator, but only first-order approximation is used in the preconditioners (to preserve uniformity of bandwidth). Though tempting in their simplicity, Dirichlet boundary conditions alone in the preconditioner were found to perform poorly in practice in mixed BC problems, as expected. The hierarchical structure of the preconditioner renders the BC mismatch between the operator and preconditioner difficult to study theoretically. The theory in [35, 39] reveals that spectral equivalence is generally lost in such BC mismatches, but only a small number of eigenvalues of the preconditioned operator may be responsible.

3.3. Comparison with Other Approaches

Before appealing to numerical experimention to illustrate the techniques presented above, we briefly compare them with other known techniques arising from similar motivations.

The field of locally uniform mesh refinement is spanned by a continuum of resolution strategies governed by clustering rules that control the size and shape of the refined subregions. Global refinement lies at one extreme and pointwise adaptive refinement at the other. As soon as the global tensor product mesh is abandoned, a host of difficult practical decisions must be made about data structures and clustering algorithms. The logic required to handle the numerous types of subgrid-subgrid interactions that can arise and to ensure the consistency of the possibly distributed data structure can be a significant impediment to efficient parallelism. It is impractical to use domain-based "horizontal" decompositions to obtain distributed parallelism if refined subgrids are allowed to span the coarse grid in a general nested fashion. Instead, parallel decompositions of general, multilevel, locally uniform composite grids should proceed by level, as argued and implemented in [37]. However, "horizontal" neighbor-neighbor interactions on a tensor-product grid of individually refined tiles are simple.

The tile algorithm requires only one grid that possesses connectivity with arbitrarily distant regions of the domain, namely, the grid of cross points. In the framework of the hierarchical basis

function technique [2, 47], we have simply a two-level hierarchy, but the higher level may be different in different subregions. Figure 5 is a one-dimensional illustration. This represents a severe condensation of the range of intermediate scales present in multilevel local uniform refinement, on which the asymptotic convergence theory is based. Tiles are much closer to being the software equivalent of the "geometry-defining processors" (GDPs) of Dewey and Patera [13]. The tile algorithm shares the philosophy of commercial structural analysis packages offering libraries of elements that an engineer can assemble in composing a domain, though comparably transparent user interfaces have yet to be written. *Unlike* most structural analysis packages, no global linear system involving all of the degrees of freedom is formed, nor is an exact Schur complement derived through the expensive process of static condensation. Rather, an iterative path to parallelism is elected.

In the latter respect, the tile algorithm is similar to the original additive Schwartz method [14] and the techniques of [6]. All three rely upon a single, coarse-domain-spanning grid. The main differences between the techniques of [6] and [14] and the tile algorithm are in the treatment of the interfacial degrees of freedom. In the additive Schwarz technique, interior problems are solved on extended overlapped subdomains, so that the interfacial degrees of freedom of one subdomain are interior points of another and thus demand no special consideration. In [6], good preconditioners for the interfacial degrees of freedom of abutting subdomains are derived theoretically for selfadjoint operators. Near optimal algebraic convergence for the refined case has been proved for both classes of algorithms in [15] and [5], respectively, for selfadjoint systems. For non-selfadjoint systems, convergence proofs for the uniformly refined case have been given in [8] and [9], respectively.

A disadvantage shared by all two-scale approaches is that the coarse grid — on which the optimal approaches perform an exact solve, and on which we also prefer one — cannot necessarily remain as coarse as one might like. In contrast, multilevel methods are not held hostage to a fine "coarse" grid. Even so, multilevel convergence estimates for non-selfadjoint operators are aided by sufficiently fine coarse grids, and complex domain geometry or "ragged" coefficients can also make a fine coarse grid desirable in practice.

General multilevel methods with a number of levels substantially larger than two maintain their optimal convergence rates at the price of increasingly complex data-dependency patterns with attendant degradation on multiprocessor architectures and intricacy of coding in practical problems. The additive or asynchronous methods [36] relieve most of the interlevel data traffic but do not obviate the need to collect data vertically across the levels at each iteration. The ability of a two-level approach to obtain convergence rates only a log factor worse than optimal is demonstrated in Section 4. Compelling overall superiority of approaches with a greater richness of scales has not been established in production parallel software. In the course of establishing it, experience on parallel computers with a two-level algorithm will be beneficial and will aid in evaluating the complex tradeoffs.

We have too little experience with the full spectrum of methods discussed above to conjecture about the sizes of the relevant constants in asymptotic complexity analyses or to provide experimental comparisons (but see [9] for a comparison of the tile algorithm with additive Schwarz on a model scalar convection-diffusion problem). It is clear, however, that the limitations of the tile algorithm are shared to some degree by the optimal methods, while the simplicity of implementation and straightforwardness of generalization are not universally shared.

4. Numerical Experiments

To illustrate the effectiveness of the tile algorithm in terms of the convergence of the iterations, and the effectiveness of the locally uniform mesh refinement in terms of the convergence of the discretization, we consider a suite of experiments.

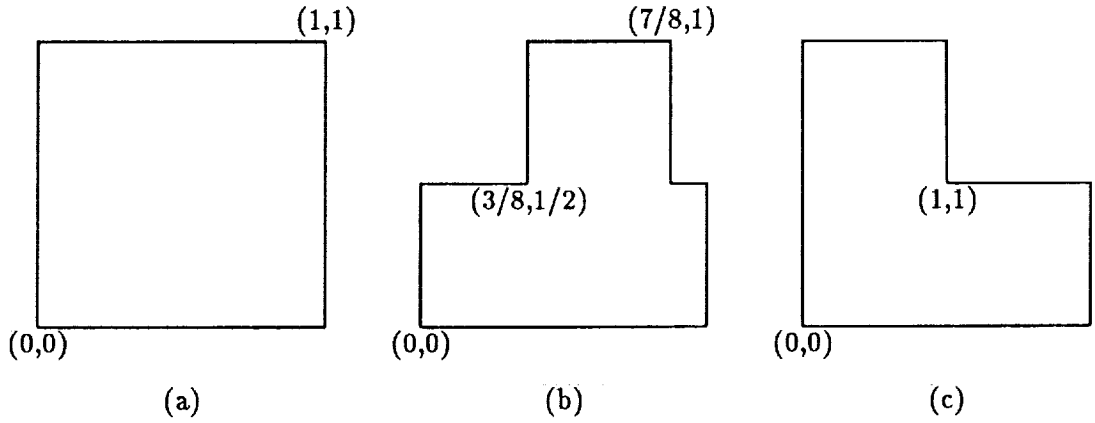


Figure 6: The three domains considered in this paper.

4.1. Model Problems

We present ten model problems, each containing a single dependent variable and two independent variables. Some of the problems below are selfadjoint and could be discretized in a symmetric manner and perhaps solved more cheaply with conjugate gradients than with GMRES. Our main interest, however, is in the more extensible formulation. In all the examples an exact solution of the continuous problem $\mathcal{L}u = f$ is specified. From this $u(x, y)$, all of the source terms f and boundary condition inhomogeneities g may be calculated. In cases where the expressions for f and g are sufficiently simple, they are written out along with the solution. The ten problems are defined over three different domains, pictured in Figure 6. Perspective surface plots of the solutions to the problems are given in Figures 7 and 8.

The first two examples, with constant coefficients and an exact solution quadratic in each independent variable, are extremely simple and possess second-order finite difference representations free from truncation errors. They are identical except for the type of boundary conditions along one side of their square domain. These problems are not candidates for mesh refinement; rather, they were chosen to illustrate the deterioration in convergence rate caused when Dirichlet boundary conditions are replaced with Neumann, and to allow controlled experimentation on the effect of mismatched boundary conditions in the preconditioner. The poor convergence of #2 using the preconditioner of #1 originally forced the decision to expand the cross-point system to include physical boundary points in the general case.

Problem #1: Pure isotropic diffusion with all Dirichlet boundaries

$$\nabla^2 u = 4$$
$$u(x, y) = x^2 + y^2$$
Dirichlet data on $\partial\Omega$
Ω = Unit square

Problem #2: Pure isotropic diffusion with a partial Neumann boundary

$$\nabla^2 u = 4$$
$$u(x, y) = x^2 + y^2$$

Dirichlet data on the three lower sides of $\partial\Omega$

$$\frac{\partial u}{\partial n}(x,1) = 2$$

Ω = Unit square

The next example is included to study orientation sensitivity of the substructuring resulting from anisotropic diffusion, for comparison with Problem 1, to which it is identical when $a = 1$. It is of further interest in that the order-of-magnitude ratio between the diffusion coefficients in the x and y directions is mathematically indistinguishable at the discrete level from an order-of-magnitude physical domain aspect ratio in an isotropic diffusion problem. Thus, the discretized version of Problem 3 covers two physical problem parameter extremes in one.

Problem #3: Anisotropic diffusion

$$\frac{\partial}{\partial x}\left(a\frac{\partial u}{\partial x}\right) + \frac{\partial^2 u}{\partial y^2} = 2(a+1)$$

$$u(x,y) = x^2 + y^2$$

$$a = 10$$

Dirichlet data on $\partial\Omega$

Ω = Unit square

The fourth example is a prototype convection-diffusion problem: a passive scalar in a plug flow that is well developed at the outflow. It is a companion problem to #2 in the sense of possessing a smooth solution with one Neumann boundary, but it is asymmetric as a result of the convection term. In that its anisotropy comes from a first- rather than second-order operator, it also complements #3.

Problem #4: Plug-flow convection-diffusion with fully developed outflow boundary

$$-\nabla^2 u + c\frac{\partial u}{\partial y} = f$$

$$u(x,y) = \sin(\pi x)\sin\left(\frac{\pi y}{2}\right)$$

$$c = 10$$

$u = 0$ on the three lower sides of $\partial\Omega$

$$\frac{\partial u}{\partial n}(x,1) = 0$$

Ω = Unit square

The next two canonical examples (from the "population" of elliptic problems in [40, 41]) bring in nonconstant coefficients, the latter in a non-selfadjoint way with Robin boundary conditions.*

Problem #5: Selfadjoint, nonconstant coefficient, Dirichlet boundaries

$$\frac{\partial}{\partial x}\left(e^{xy}\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(e^{-xy}\frac{\partial u}{\partial y}\right) - \frac{u}{1+x+y} = f$$

$$u(x,y) = e^{xy}\sin(\pi x)\sin(\pi y)$$

$$u = 0 \text{ on } \partial\Omega$$

Ω = Unit square

*The more widely available reference [28] contains an identical listing of Problem 5 and a similar but not identical version of #6. A typographical error in the latter renders it ill-posed.

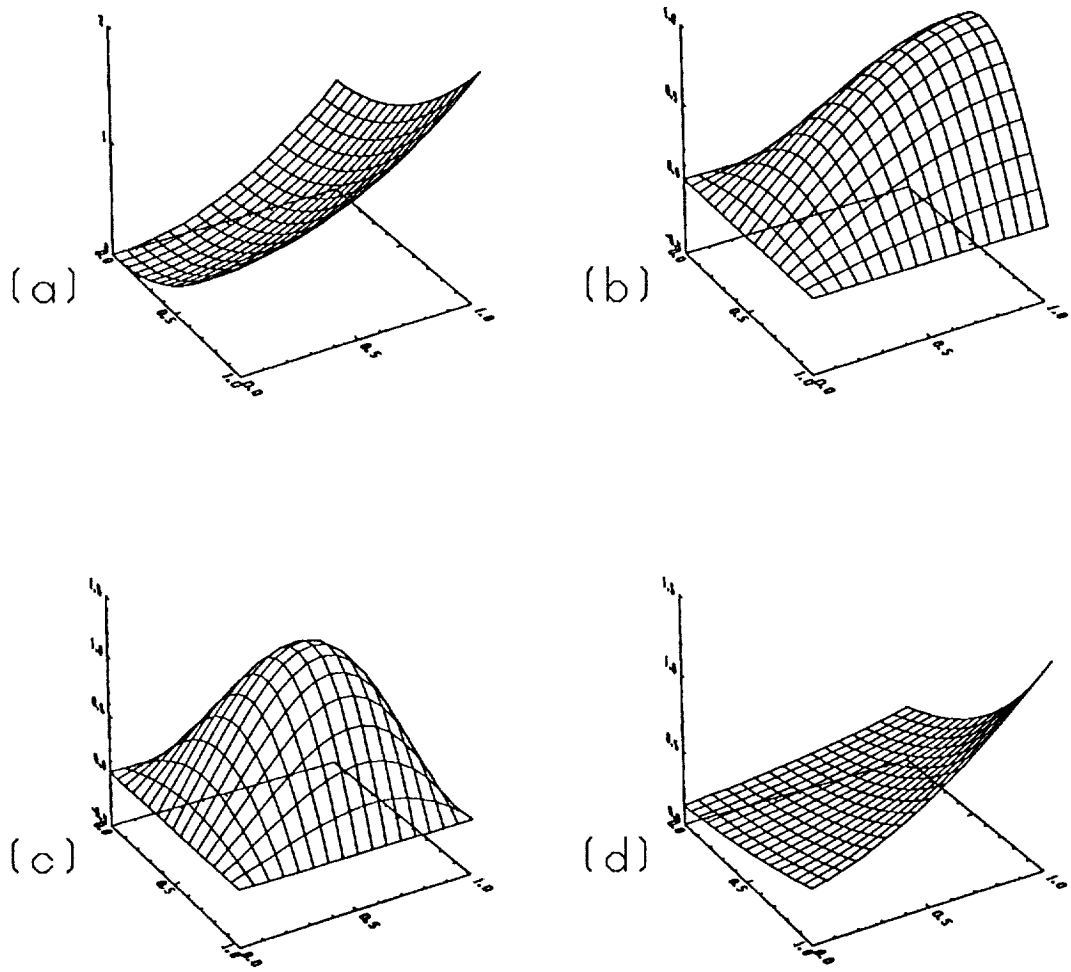


Figure 7: Surface plots of the test problem solutions: (a) #1–3, (b) #4, (c) #5, (d) #6.

Problem #6: Non-selfadjoint, nonconstant coefficient, Robin boundaries

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial}{\partial y}\left((1+y^2)\frac{\partial u}{\partial y}\right) - \frac{\partial u}{\partial x} - (1+2y+y^2)\frac{\partial u}{\partial y} = f$$

$$u(x,y) = 0.135(e^{x+y} + (x^2-x)^2\log(1+y^2))$$

$$u - \frac{\partial u}{\partial n} = g \text{ on } \partial\Omega$$

$$\Omega = \text{Unit square}$$

The derivative is the outward normal.

The seventh example, from [4, 30], is on an irregularly shaped domain with reentrant corners, but possesses a smooth solution. It emphasizes how an irregular domain may force a minimum granularity upon a tessellation comprising congruent tiles. For the problem at hand, however, the minimum granularity is near the ideal one.

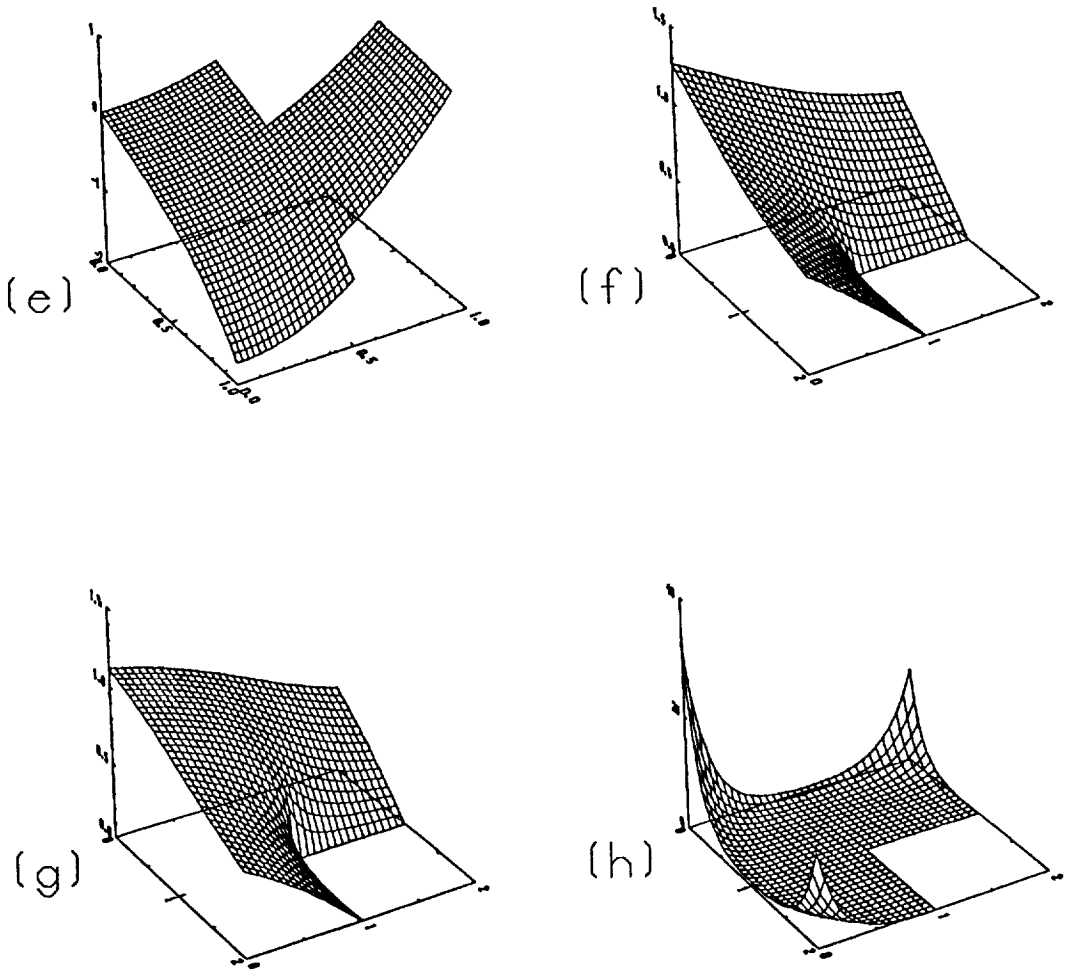


Figure 8: Surface plots of the test problem solutions: (e) #7, (f) #8, (g) #9, (h) #10.

Problem #7: T-shaped domain

$$\nabla^2 u = 4 - 2\cos(y)e^x$$
$$u(x,y) = x^2 + y^2 - xe^x\cos(y)$$
Dirichlet data on $\partial\Omega$
Ω = T-shaped region

The last three examples are obtained by taking three different values of the convection — respectively, $c = 0$, $c = -1$, and $c = 10$ — in the convection-diffusion problem below.

Problems #8–10: Cylindrically separable reentrant corner convection-diffusion problem

$$-\nabla^2 u + \frac{c}{r}\frac{\partial u}{\partial r} = 0$$

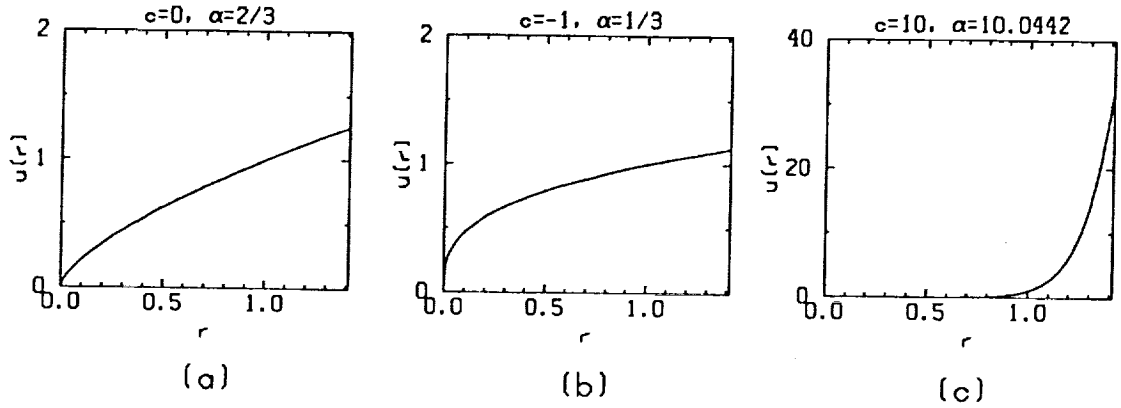


Figure 9: Cross section of $u(r)$ along the symmetry axis: (a) Problem #8, pure diffusion, nondifferentiable at $r = 0$; (b) Problem #9, convective inflow, strengthening the singularity; (c) Problem #10, convective outflow, eliminating the singularity.

$$u(x,y) = r^{\alpha} \sin\left(\frac{2}{3}\left(\theta - \frac{1}{2}\pi\right)\right)$$

$$\text{where } r = \sqrt{(x-1)^2 + (y-1)^2}$$

$$\text{and } \theta = \arg((x-1) + i(y-1)),\ 0 \le \theta < 2\pi$$

$$\text{Dirichlet data on } \partial\Omega$$

$$\Omega = \text{L-shaped region}$$

The first of these corresponds to pure diffusion, and the second and third to convection in towards the reentrant corner and away from it, respectively, at a rate inversely proportional to the radius. The respective values of the radial eigenfunction exponent α are $\frac{2}{3}$, $\frac{1}{3}$, and approximately 10.04, from the Euler equation formula $\alpha = \left[c + \sqrt{c^2 + \frac{16}{9}}\right]/2$. Figure 9 displays $u(r)$ along the ray $\theta = \frac{5\pi}{4}$, which is the symmetry axis of the three L-shaped problems. The first two solutions of this trio lack derivatives at the reentrant corner. The last is everywhere twice differentiable, but the solution is characterized by steep variation in the three *non*reentrant corner regions, where $r > 1$. Local mesh refinement is critical to improving the accuracy of a finite difference solution. In [24] we show the complementary benefit of rediscretization of the tiles surrounding the reentrant corner in Problems 8 and 9 to fit the discrete solution to the known power-law radial dependence of the singular exact solution (see the problem statements above). Rather than making the customary Taylor series assumptions, we take $u(r) = u_0 + ar^p + br^{2p}$, where p is derivable from a local analysis.

4.2. Parameters Studied

Four categories of experiments are reported. First, a two-dimensional parameter space consisting of coarse grid resolution and overall (uniform) resolution is explored by numerical experiment for each problem. The goal of these experiments is the evaluation of the convergence of the algorithm, in terms of iteration count and execution time, over a range of resolutions for comparison with a back-of-the-envelope complexity analysis in Section 4.3 and related theory in Section 4.4. No adaptive refinement is performed.

Another set of experiments is performed on Problems 8–10 only with the goal of evaluating the economy of the locally uniform refinement technique. We show that LUMR is capable of significant CPU and memory savings with no sacrifice of accuracy relative to uniform refinement.

In a third set of experiments, the effect of orientation for nonunit-aspect ratio tiles is investigated. The limiting case of stripwise decompositions shows how physical anisotropies can be exploited in the decomposition for improved convergence.

Finally, we compare the domain-decomposed preconditioner of this paper with some popular global preconditioners and with the topologically related direct solve using a nested dissection ordering [21].

Additional studies, including modular replacement of $\tilde{A}_I$ or B_B with some of the alternatives listed in Section 2, are available in [24]. Use of two different orders of discretization in A and B is explored in [33]. (This approach loses the identity block in (2.3) but delivers higher-order upwinding while preserving monotonicity in the preconditioner.) In this study, we simply use $\tilde{A}_I = A_I$, $\tilde{A}_{IB} = A_{IB}$, and $\tilde{A}_{BC} = A_{BC}$ and derive B_B from the tangential terms of the differential operator.

The timings given below are from a SPARCserver 390 with 64-bit reals. The code was written in C except for low-level Fortran kernels, such as factoring or solving linear systems entirely resident on one processor. *Relative* comparisons of CPU times for alternative formulations of the same problem executed in the same hardware and software environment are an important part of our results. It should be borne in mind while studying the results that different organizations of the code and different compiler capabilities can account for large variations in execution times across architectures and software releases; therefore, *absolute* execution times are not very meaningful. We have run the same experiments (or representative subsets, to the extent supported by memory) in scalar mode on seven other Unix machines and find that even the *proportions* of time spent in factorization and solution phases may vary widely between machines. In spite of this, there are surprisingly few shifts in the overall performance rankings of alternative decompositions. In other words, while the timings in the tables are far from machine independent, the conclusions based thereon *are*, until parallelism enters the picture.

4.3. Convergence as a Function of Coarse Grid Granularity

To test coarse-grid granularity over an interesting range, we fix the finest mesh spacing at $h^{-1} = 128$ (relative to the total length of the domain, whether that be 1 in the problems posed on the unit square or 2 in the problems on the L-shaped domain) and investigate the tradeoff between numbers of tiles and points per tile, as shown in Tables 1 and 2 and plotted in Figure 10. The mesh is identical and uniform for all runs in these tables (with the obvious exception that pieces of the circumscribing square are missing from it in Problems 7–10, whose columns therefore lack entries at the coarsest tile subdivisions). The convergence criterion is a relative reduction in residual of five orders of magnitude. Throughout these studies we use an initial iterate of zero. Table 1 shows that the iteration count peaks in the middle of the granularity range, at 4 or 8 tiles per side, and decreases to 1 in either degenerate limit of one tile per domain or one tile per point (not shown), where a global direct solve results.

Table 2 shows the deceptiveness of iteration count alone as a measure of overall performance. In execution time, the extreme runs, representing few-domain cases, suffer as a result of the high cost per iteration, even though the number of iterations required is very small. This table is a profound illustration of an earlier version of [11], entitled *Domain Decomposition Beneficial Even Sequentially.* The most favorable total *sequential* execution times are found for multidomain cases at 16 or 32 tiles per side.

The factorization of the banded matrix in the single subdomain case is the dominant contribution to the overall time. In Problems 1–6, over six minutes are spent doing the factorization alone. A similar penalty would accrue in an attempt to do direct solves on a very fine "coarse" grid, in which each tile contains just one point. However, this second peak is not visible since the table is truncated below tile sizes of $H/h = 4$. Even in modest-sized two-dimensional problems,

H^{-1}	H/h	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
1	128	1	1	1	1	1	1	–	–	–	–
2	64	10	14	18	25	26	17	–	12	11	4
4	32	11	15	24	25	32	21	–	15	16	15
8	16	9	12	25	21	29	16	11	14	15	16
16	8	7	10	22	18	26	12	10	11	12	13
32	4	6	7	15	14	21	7	8	8	9	8

Table 1: Iteration count as a function of number of tiles per side of the circumscribing square, H^{-1}, and number of mesh points along a tile side, H/h, at constant refinement parameter, $h^{-1} = 128$, for a reduction in the initial residual of 10^{-5}.

H^{-1}	H/h	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
1	128	371.	376.	371.	373.	371.	375.	–	–	–	–
2	64	109.	118.	127.	144.	145.	125.	–	86.1	84.2	73.0
4	32	34.7	39.7	51.9	53.3	62.4	47.9	–	30.3	31.4	30.5
8	16	12.0	14.4	26.2	22.5	29.8	17.8	10.2	11.9	12.4	13.1
16	8	5.5	8.0	18.3	14.4	22.1	9.6	5.7	6.0	6.7	7.2
32	4	6.8	7.9	17.9	16.3	27.2	7.9	6.5	6.4	7.1	6.4

Table 2: Total execution time (sec), including both preconditioner factorization and GMRES iteration, as a function of number of tiles per unit length, H^{-1}, and number of mesh points along a tile side, H/h, at constant $h^{-1} = 128$, for a reduction in the initial residual of 10^{-5}.

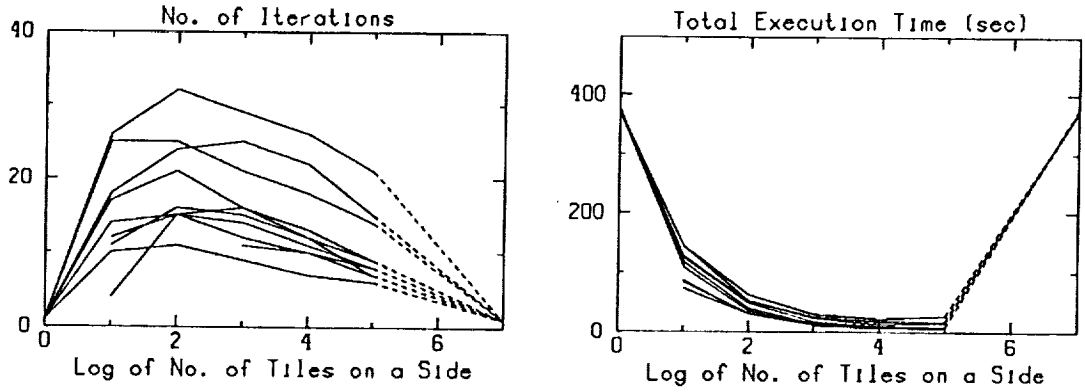


Figure 10: Plots of Tables 1 and 2 (Problems 1–10 superposed), illustrating that the minimum execution time of the serial algorithm occurs near $H^{-1} = 16$ tiles on a side, though the maximum iteration count occurs near this granularity. (The dashed portions of the curves are extrapolated beyond the data of the tables.)

direct solves on the undecomposed domain are inefficient relative to decomposition-preconditioned GMRES. Of course, there are many alternatives to direct solves for solving a smooth elliptic equation discretized on a tensor-product grid problem on a uniprocessor, some of which are considered

H^{-1}	h^{-1}	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
2	16	6	9	11	11	12	11	NA	6	6	3
4	32	9	12	17	15	19	17	NA	12	12	10
8	64	9	11	22	18	23	15	10	12	13	14
16	128	7	10	22	18	26	12	10	11	12	13

Table 3: Iteration count as a function of number of tiles per side of circumscribing square, H^{-1}, and refinement parameter, h^{-1}, at constant number of mesh points along a tile side, $H/h = 8$, for a reduction in the initial residual of 10^{-5}.

in Section 4.7, but most are not coded or parallelized as cleanly as domain-decomposed Krylov iteration.

The behavior in Table 2 can be understood with reference to back-of-the-envelope complexity estimates for the solution and factorization operators of the preconditioner. We observe that there are $\mathcal{O}(H^{-2})$ cross points, interfaces, and interiors. Naturally ordered banded direct factorizations and solves require $\mathcal{O}(Nb^2)$ and $\mathcal{O}(Nb)$ operators, respectively, where N is the number of unknowns and b the bandwidth. For the cross-point system, $N \approx H^{-2}$ and $b \approx H^{-1}$; for the interfaces, $N = H/h$ and $b = 1$; and for the subdomain interiors, $N = (H/h)^2$ and $b = H/h$. Thus, the interface operation counts are always asymptotically subdominant and can be omitted in the following. From choosing the larger of the cross-point and interior complexities, we see that factorization costs $\max\{\mathcal{O}(H^{-4}), \mathcal{O}(H^2h^{-4})\}$ and solves cost $\max\{\mathcal{O}(H^{-3}), \mathcal{O}(Hh^{-3})\}$. The first term grows with H^{-1} and the second decays with it. Quick calculations reveal that (to the resolution of the table) the minima for both factorization and solve costs occur at or between $H^{-1} = 16$ and 32 when $h^{-1} = 128$. The tendency of buffer overhead, neglected in these estimates, is to favor a slightly smaller number of tiles per side than thus estimated. It is important to note that the memory requirements follow the solve complexities above. Thus, for a fixed memory size, an intermediate cross-point grid granularity accommodates the largest problem in core. Of course, all of these per iteration complexity estimates must be redone when the preconditioner blocks are other than banded direct solves.

4.4. Convergence as a Function of Refinement

In contrast to the preceding section, we here investigate iteration count as a function of overall resolution, for a fixed number of subintervals per tile. The results are shown in Table 3. The global mesh grows in refinement from 16 to 128 while the number of points per tile remains constant at 8. Thus, the fine grid in the last row of Table 3 corresponds to the $H^{-1} = 16$ row of the earlier tables. In spite of the fact that the truncation error improves with h^{-2} in some of these problems, we impose a constant convergence tolerance of 10^{-5} on the tests in the upper rows of Table 3, in order to focus on the algebraic convergence alone.

With the minor exception of #5, which has not quite reached its iteration maximum at 16 tiles per side, the experiments suggest that the iteration count is bounded as resolution increases at constant H/h. In over half of the cases, the finest mesh results are even relatively *better* than the immediately preceding coarser ones. This fact is not surprising since there is a price for this favorable iteration count when H/h is held constant and h^{-1} is increased, namely, a larger cross-point system. The theory for conjugate gradient iteration for selfadjoint problems [6] and for GMRES iteration for non-selfadjoint problems [9] contains similar results for abutting domains, namely, constant upper bounds on the iteration count for constant H/h.

As representative convergence histories, we present Figure 11 which follows the residual reduction over five orders of magnitude, and the time versus iteration count history for Problems 1 and 2. In the latter plots, the quadratic term in the GMRES work estimate (that comes from the

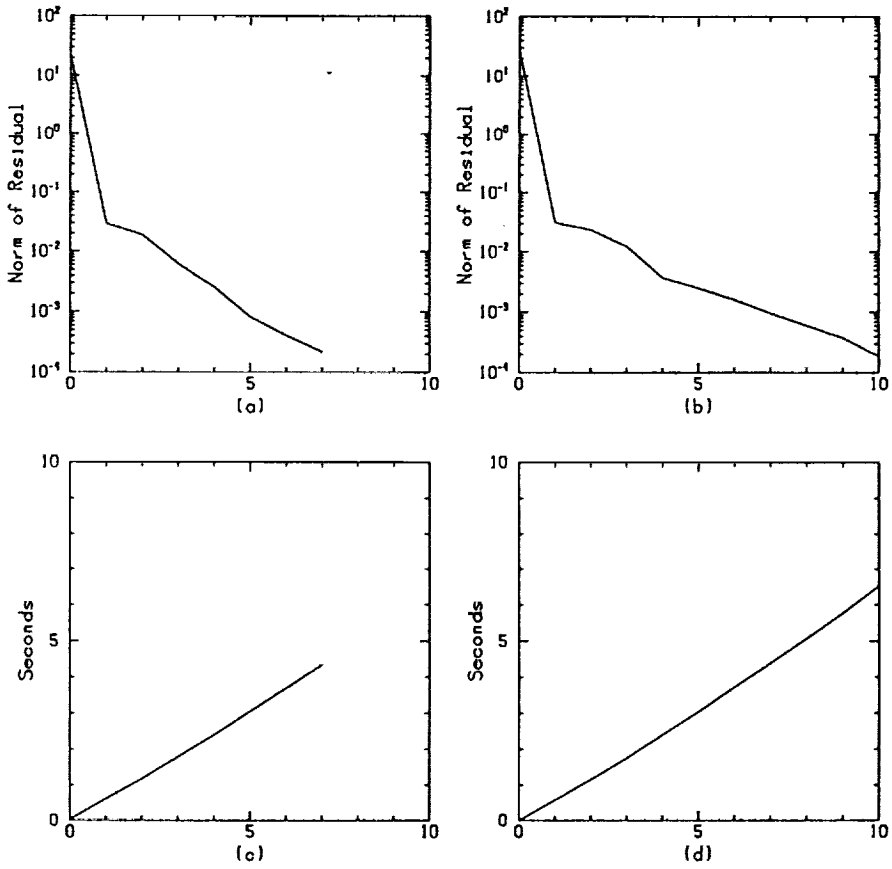


Figure 11: Convergence histories for Problems 1 and 2, for $H^{-1} = 16$, $H/h = 8$, $h^{-1} = 128$. (a) and (b) show the normalized Euclidean norm of the residual versus iteration count, and (c) and (d) show time versus iteration count.

need to orthogonalize each iterate over a subspace whose size grows linearly in iteration count) is almost invisible. This is due to the exploitation of the identity row in (2.3). This pair of figures also illustrates the poorer conditioning of Neumann problems, since the initial iterates and the solutions converged to are identical, and so are the operators except for one Neumann boundary segment.

4.5. Economies of Local Mesh Refinement

Problems 8–10 can be used to illustrate the well-known benefits of local uniform mesh refinement in elliptic problems: comparable accuracy in considerably fewer operations, compared with global uniform refinement. We solve these problems at effective refinement levels of $h^{-1} = 32$, 64, 128, and 256, based on the global grid, but perform both global and local refinements for comparison where possible. (The finest global refinement does not fit into the memory available, which is, of course, another of the main motivations for LUMR, along with execution time savings.) All of these computations were made with a reduction in the algebraic residual of 10^{-8}, so that the measure of the truncation error would not be contaminated. The choice of where to refine is made manually.

Tables 4 through 6 compare global refinement results on the left, and local on the right. The

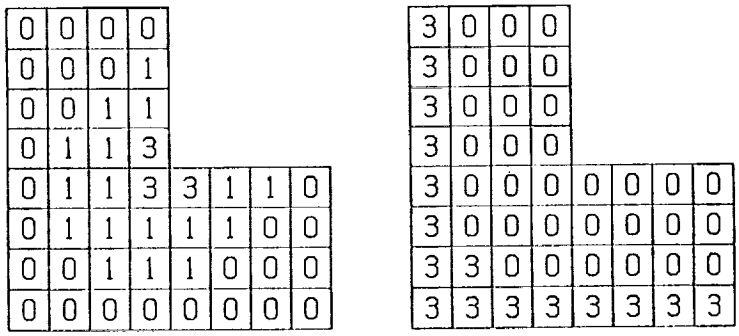


Figure 12: Refinement levels, indicated by the logarithm of the refinement ratio. The maximum (third level) local uniform refinements are shown: (a) Problems 8 and 9, (b) Problem 10. In second-level tests, all tiles showing "3" are set to "2". In first-level tests, these are further reduced to "1". In zeroth-level refinement, all tiles are set to "0", which here corresponds to $H/h = 4$.

		Global				Local				Ratio
h^{-1}	H/h	N_G	e_G	I_G	T_G	N_L	e_L	I_L	T_L	T_G/T_L
32	4	833	1.30(-2)	18	2.6	833	1.30(-2)	18	2.6	1.00
64	8	3201	8.30(-3)	22	8.1	1817	8.30(-3)	22	5.5	1.47
128	16	12545	5.25(-3)	26	48.1	2409	5.26(-3)	23	7.9	6.08
256	32	–	–	–	–	4745	3.33(-3)	28	31.8	–

Table 4: Number of unknowns N, sup-norm of the error e, iteration count I, and execution time T (sec) for Problem 8 (reentrant corner, pure diffusion), globally and locally refined, along with execution time ratios, for a reduction in the initial residual of 10^{-8}.

local refinement is as illustrated in Figure 12. Each set of columns lists the number of unknowns, the sup-norm of the error, the number of iterations to reduce the discrete residual by 8 orders of magnitude, and the total execution time thus required. The right-most column gives the execution time ratios for each refinement level. Memory use ratios can also be estimated from the tile structure of the discrete problem, but the present code records no explicit allocation measurements. All entries share a constant value of $H^{-1} = 8$ in order to fix in space regions of enhanced refinement that do not shrink as h does. Therefore, the "global" iteration columns of Tables 4–6 incidentally provide a constant-H traverse through convergence rate parameter space, complementary to Table 1 (a constant-h traverse) and Table 3 (a constant-H/h traverse).

The linear increases of iteration count with each doubling of global refinement in the selfadjoint problem in Table 4 and the nearly selfadjoint problem in Table 5 are consistent with a logarithmic growth in conditioning with h^{-1}. The locally refined examples likewise worsen mildly in conditioning with h^{-1} when H is held constant, but the CPU time advantage, (T_G/T_L), of local refinement increases with h^{-1} overall. Comparing iteration counts of globally and locally refined problems at the same effective refinement shows that the often drastically smaller number of unknowns in the

		Global				Local				Ratio
h^{-1}	H/h	N_G	e_G	I_G	T_G	N_L	e_L	I_L	T_L	T_G/T_L
32	4	833	6.97(-2)	18	2.5	833	6.97(-2)	18	2.5	1.00
64	8	3201	5.65(-2)	23	8.7	1817	5.66(-2)	23	5.8	1.50
128	16	12545	4.53(-2)	28	51.3	2409	4.58(-2)	25	8.9	5.76
256	32	–	–	–	–	4745	3.67(-2)	28	31.9	–

Table 5: Same as Table 4, except for Problem 9 (convective inflow).

		Global				Local				Ratio
h^{-1}	H/h	N_G	e_G	I_G	T_G	N_L	e_L	I_L	T_L	T_G/T_L
32	4	833	7.35(-1)	19	2.9	833	7.35(-1)	19	2.4	1.00
64	8	3201	4.15(-1)	23	8.8	1609	4.30(-1)	22	5.1	1.73
128	16	12545	2.19(-1)	29	55.6	4697	2.40(-1)	27	20.1	2.77
256	32	–	–	–	–	17017	1.98(-1)	34	170.2	–

Table 6: Same as Table 4, except for Problem 10 (convective outflow). The error values here appear large but are, in fact, small relative to the size of the solution (recall Figure 9c).

latter does not much affect convergence. This observation leads to the hypothesis that in the case of variously refined tiles, (H/h_{finest}) is the convergence-controlling parameter, with the details of the tile size distribution important only in estimating the work per iteration.

The sup-norm of the error shows sublinear improvement in h in Problems 8 and 9, as one expects with nondifferentiable solutions; and, though the solution of Problem 10 is smooth, the first-order accurate treatment of convection leaves its signature instead.

4.6. Anisotropic Decompositions

Throughout the foregoing, we have considered decompositions of the problem domain into uniform square tiles exclusively. More general decompositions are possible and should often be considered. Varying the aspect ratio and the orientation of tiles can lead to significant variations in convergence rate in anisotropic problems. Problems #1–4 of the test suite contain a sufficient variety of operators and boundary conditions to illustrate this point. Table 7, based on these four problems, provides a link between the boxwise decompositions studied in this report and the stripwise decompositions employed in many of our earlier studies, such as [32]. Four different decompositions are considered in this table: vertical strips, horizontal strips, a boxwise decomposition with the same number of tiles as the strip cases, and a boxwise decomposition whose tiles have the same bandwidth as that of the most compact natural ordering of the strip cases. These decompositions are shown in Figure 13. In every case, the mesh spacing is held constant at $h^{-1} = 128$; thus these problems contain 16,641 discrete unknowns.

Among the two boxwise decompositions, the finer is always closer to the optimum found earlier in Table 1 for iteration count and in Table 2 for execution time.

Contrasting the boxwise and stripwise decompositions, we note that for isotropic Problems 1 and 2 the isotropic (boxwise) decompositions lead to significantly better iteration counts than the nonisotropic (stripwise) decompositions. The coarser boxwise decomposition nevertheless leads to poorer execution times than either stripwise decomposition because of the large bandwidth of its tiles resulting in large factorization costs in setting up the interior solves of the preconditioner. The finer boxwise decomposition yields the best execution times.

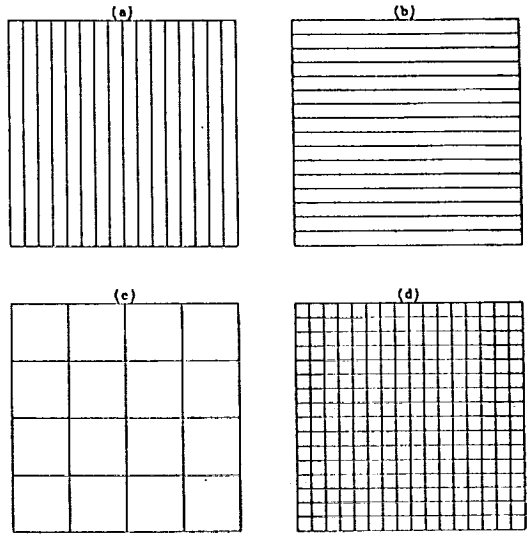


Figure 13: The four decompositions tested in Table 7: (a) vertical strips, (b) horizontal strips, (c) "large" boxes with the same granularity as the strips, (d) "small" boxes with the same bandwidth as the strips.

			#1		#2		#3		#4	
Case	H_x^{-1}	H_y^{-1}	I	T	I	T	I	T	I	T
a	16	1	20	13.7	29	20.5	52	40.5	4	3.5
b	1	16	20	13.7	23	15.9	12	8.4	16	10.9
c	4	4	11	34.7	15	39.7	24	51.9	25	53.3
d	16	16	7	5.5	10	8.0	22	18.3	18	14.4

Table 7: Iteration count I and total execution time T (sec) as a function of the tessellation parameters H_x^{-1} and H_y^{-1}, at constant mesh parameter $h^{-1} = 128$, for a reduction in the initial residual of 10^{-5}.

For the nonisotropic Problems 3 and 4, one or both of the nonisotropic decompositions is superior in both iteration count and execution time to both of the isotropic decompositions. In Problem 3, we note the major advantage of handling the strong x-directional diffusive coupling as implicitly as possible in the preconditioner when few (here 16) subdomains are employed. In Problem 4, where the diffusive part of the operator is isotropic, the decomposition that is aligned with the strong convection is superior. This is related to the relatively poorer performance of the tangential interface preconditioner in problems where the convection is normal to the interface rather than tangential to it [12]. In spite of the poor representation of the convection in the interface blocks for Problem 4, case (b), the "wrong" stripwise decomposition is still slightly superior to the best boxwise decomposition. Though the boxwise decomposition convergence rates are asymptotically superior to the stripwise decomposition convergence rates (see the theoretical arguments summarized in [30]), the crossover point is evidently strongly influenced by the physics of the problem.

4.7. Comparison with Undecomposed Preconditioners

Tables 1 and 2 produced the observation that among preconditioners employing direct banded

Method	#1	#2	#3	#4	#5	#6
GMRES(90)/ILU(0)	73	108	80	82	99	>
GMRES(90)/MILU(0)	22	77	19	147	39	>
GMRES(90)/ILU(1)	45	58	61	51	59	160
GMRES(5)/ILU(0)	351	312	227	>	>	>
GMRES(5)/MILU(0)	27	141	23	>	57	>
GMRES(5)/ILU(1)	139	150	140	213	244	>
Direct	1	1	1	1	1	1
GMRES(90)/DD	7	10	22	18	26	12
GMRES(5)/DD	8	10	28	25	39	12

Table 8: Iteration counts for problems #1–6 for seven different algorithmic combinations at a mesh parameter $h^{-1} = 128$, for a reduction in the initial residual of 10^{-5}. ">" signifies more than 500 iterations.

factorizations for both the cross-point system and the subdomain interiors, a tessellation of intermediate granularity is much superior to one at either coarse or fine extremes. In other words, domain decomposition-preconditioned GMRES methods are superior to bandsolvers even on sequential computers and even in two dimensions. It is of interest to attempt to strengthen such a statement by comparing domain decomposition-preconditioned GMRES iteration with other candidate solvers in the sequential, two-dimensional context. For this purpose, a direct sparse matrix solver and three popular incomplete factorizations have been implemented as alternative subdomain interior preconditioners and compared with the domain-decomposed preconditioner on the first six problems of the test suite.

Table 8 lists the iteration counts and Table 9 the execution times for nine different solution algorithms callable from the same code used to generate all previous tables. (The global domain solvers contain just one tile.)

Six of these solvers are iterative methods based on GMRES and global preconditioners of the incomplete factorization type, tested in two sets of three each. In each set, we test ILU(0), ILU(1), and MILU(0) [16, 38], where the integer in parentheses denotes the number of diagonals of extra fill-in retained adjacent to the original five-diagonal structure of the discrete operator [46]. In the first set, the maximum size of the Krylov subspace used in GMRES is 90; in the second set, the maximum Krylov subspace has dimension 5. In a majority of cases, the globally preconditioned GMRES iteration converges in fewer than 90 iterations; thus, the first set consists mainly of full GMRES convergence results. In practical applications, restarted GMRES is often used to conserve memory or defeat the quadratic term in the GMRES work estimate that arises from orthogonalization over an ever-expanding Krylov subspace. GMRES(k) denotes a restart after k steps.

One of the solvers is a direct method, the Yale Sparse Matrix Package [17] using a global nested-dissection ordering (rather than the minimum degree ordering provided with YSMP), which naturally converges in one step.

Finally, we test two domain-decomposed GMRES algorithms based on a 16×16 array of 8×8 tiles. Both are slight variants of the algorithm used in preceding sections in which the bandsolver is replaced with the nested-dissection sparse solver. Full GMRES and GMRES(5) are considered.

Comparing first the convergence rates of the various global preconditioners, we observe that in the diffusively dominated problems with Dirichlet boundary conditions (#1,3,5) the fill-capturing modified incomplete factorization MILU(0) is much superior to ILU(0) and ILU(1). The existence of a non-Dirichlet boundary segment weakens MILU (boundary conditions are the only difference

Method	#1	#2	#3	#4	#5	#6
GMRES(90)/ILU(0)	121.	191.	142.	149.	183.	–
GMRES(90)/MILU(0)	16.7	133.	13.5	256.	41.2	–
GMRES(90)/ILU(1)	54.2	83.1	90.1	66.7	85.4	297.
GMRES(5)/ILU(0)	195.	173.	126.	–	–	–
GMRES(5)/MILU(0)	14.9	78.6	*12.6*	–	31.5	–
GMRES(5)/ILU(1)	82.8	90.7	84.3	128.	136.	–
Direct/Nest. Diss.	71.2	70.1	70.1	70.1	70.0	71.6
GMRES(90)/DD/Nest. Diss.	*7.1*	*9.9*	23.8	*18.7*	*29.7*	*12.0*
GMRES(5)/DD/Nest. Diss.	8.2	10.4	25.0	23.2	34.2	12.0

Table 9: Execution times (sec) for problems #1–6 for ten different algorithmic combinations at a mesh parameter $h^{-1} = 128$, for a reduction in the initial residual of 10^{-5}. The best time in each column is italicized.

between #2 and #1), and the presence of convection weakens it substantially (#4,6). As expected, ILU(1) uniformly requires fewer iterations than ILU(0) in these tests, and this convergence rate advantage translates into an execution time advantage even after the marginally higher cost of the ILU(1) preconditioner is taken into account. Experience with ILU(l) shows a law of diminishing returns as l increases beyond a fairly small problem-dependent value. The tests with GMRES(5) show how the higher iteration counts of a restarted method often translate into lower execution times for well-conditioned problems, but how poorly conditioned problems may fail to converge with too small a Krylov subspace.

Having noted the strong degree of problem dependence in the selection of the best global preconditioner, we note that this problem dependence extends to the relative ranking of globally preconditioned GMRES and the direct sparse nested dissection factorization. In terms of execution time, the nested dissection method loses out to the best global iterative method, GMRES(5)/MILU(0), in the odd-numbered problems, is close to the best global iterative method, GMRES(90)/ILU(1) in Problem 4, and beats all globally preconditioned methods in Problems 2 and 6.

Comparison of the nested dissection rows of Table 9 with the rows of Table 2 at corresponding granularity reveals, as expected, that the sparse direct subdomain solvers run faster than the banded direct subdomain solvers on large problems (approximately 70 sec versus approximately 370 sec on 128 × 128 tiles) and slower than bandsolvers on small ones (by approximately 20-30% on 8 × 8 tiles). The latter observation justifies our use of bandsolvers to perform the A_I^{-1} solves in the preconditioner throughout the majority of this report, where the focus is on relatively fine granularity.

Finally and most significantly, we observe that domain decomposition-preconditioned GMRES always beats the direct method, and it beats the best globally preconditioned method in all problems except for #3, for which good preconditioners of both global and domain-decomposed varieties can be found. Overall, it is the fastest executing method and performs reliably and evenly over the range of problems considered. It is a compelling serial algorithm even apart from the virtues of modularity and adaptability.

5. Conclusions and Future Directions

Experiments on a variety of model problems demonstrate that a two-level domain decomposition algorithm with a single global coarse grid provides effective convergence and convenient

refinement and permits a data structure amenable to parallel and vector implementations, as summarized in closing below. Although often motivated by parallelization, domain decomposition also yields runtime and memory use benefits as a sequential algorithm. Relative to traditional global preconditioners, domain-decomposed preconditioners can dramatically improve convergence rates. Furthermore, the simple structure of individual blocks of the domain-decomposed preconditioner means that new applications are found for the "standard solvers" in conventional software libraries. The traditional economies of local uniform mesh refinement can be incorporated into the domain decomposition framework at the small price of interface handlers with conditionals for refinement differences between adjacent subdomains. Because of the highly modular nature of a tile-oriented domain decomposition code, custom discretizations for certain classes of singularities may be archived into applications libraries for reuse. For example, a discretization tailored to the corner singularities in Problems 8 and 9 was easy to add by creating three different rotations of a special tile in [24]. In short, software engineering is a major motivation for the restricted class of algorithms explored here.

The applicable problem class is greater than the present examples indicate; for instance, the tile algorithm has been extended to multiple-dependent variable cases. A two-independent-variable streamfunction-vorticity formulation of the incompressible Navier-Stokes equations is considered in [26, 27]. The nonlinearity in this problem is handled by a Newton method wrapped around the domain-decomposed linear solver. The entire nonlinear code has been parallelized on shared- and distributed-memory machines, and the linear and nonlinear portions are comparable in their parallel efficiencies (which vary in the usual way from arbitrarily good to arbitrarily bad, depending upon problem size relative to number of processors).

Extension of the tile algorithm to a brick algorithm in three-dimensional problems is conceptually straightforward. The software engineering motivation for restriction to a tensor-product grid of substructure vertices is even more compelling in three dimensions than it is in two. One new feature is the presence of two-dimensional interfaces, upon which preconditioner blocks could be constructed by dropping normal derivative terms, by analogy with one-dimensional interfaces in the plane. The effectiveness of this straightforward extension is not demonstrated at present. For the theoretically endowed selfadjoint case it is known that the condition number of the hierarchically preconditioned system grows like the first power of (H/h), not merely like its logarithm. A discussion and some alternatives are presented in [43].

The tile algorithm is amenable to vectorization in either of two ways. The regular operation sequences on the tensor-product subgrid arrays are precisely the type for which vectorizing compilers were conceived. The vector lengths depend on the precise form of solvers used in the preconditioner but would tend to be rather small for the rows of individual 8×8 or 16×16 tiles found best in the two-dimensional applications above. An alternative form of vectorization can be realized by grouping together all tiles of a given (discrete) size and shape and operating in lock step on corresponding elements in each tile, assuming an identical solver is applied to each. A vector in this approach consists of the i^{th} element from each of the subdomains. Thus, 8×8 arrays of tiles deliver optimal processing rates for machines with a vector length of 64.

Parallelization requires attention to the load balancer/mapper [26] and also to the coarse grid solve in the preconditioner [23]. The main disadvantage of the two-level algorithm in the parallel context is that the choice of coarse grid granularity is more of an "overdetermined" problem than in serial. Communication cost per iteration and convergence properties potentially inveigh against the lower bounds on the number of tiles imposed by domain geometry, solution and coefficient roughness, and parallel load balance. The key determination for future applications of the tile methodology will be whether this overdetermination is "consistent" in practice. Inasmuch as the

early examples are representative of one or two dependent variable problems, and parallel communication costs generally comprise a *relatively* smaller proportion of the total work in coupled multicomponent problems, there are substantial grounds for optimism that this will be the case.

Acknowledgments

We express our deep appreciation to Dr. Xiao-Chuan Cai for his influence on the refinement of the tile algorithm through his mathematical insight and through his experience in using the code and in adapting it to additive Schwarz-type preconditioning. We are also indebted to two anonymous referees for constructively critical comments on an early version of this manuscript.

References

[1] I. Babuška, J. Chandra, and J. Flaherty eds., *Adaptive Computational Methods for Partial Differential Equations,* SIAM, Philadelphia, 1983.

[2] R. E. Bank and H. Yserentant, Some Remarks on the Hierarchical Basis Multigrid Method, T. F. Chan, R. Glowinski, J. Periaux, and O. Widlund eds., *Second International Symposium on Domain Decomposition Methods,* SIAM, Philadelphia, 1989, pp. 140–146.

[3] M. J. Berger and J. Oliger, *Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations,* J. Comp. Phys., 53(1984), pp. 484–512.

[4] P. E. Bjorstad and O. B. Widlund, *Iterative Methods for the Solution of Elliptic Problems on Regions Partitioned into Substructures,* SIAM J. Num. Anal., 23(1986), pp. 1097–1120.

[5] J. H. Bramble, R. E. Ewing, J. E. Pasciak, and A. H. Schatz, *A Preconditioning Technique for the Efficient Solution of Problems with Local Grid Refinement,* Comp. Meth. Appl. Mech. Eng., 67(1988), pp. 149–159.

[6] J. H. Bramble, J. E. Pasciak, and A. H. Schatz, *The Construction of Preconditioners for Elliptic Problems by Substructuring, I,* Math. Comp., 47(1986), pp. 103–134.

[7] A. Brandt, Multi-level Adaptive Techniques (MLAT) for Fast Numerical Solution to Boundary-Value Problems, H. Cabannes and R. R. Temam eds., *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics, Lecture Notes in Physics, 18,* Springer-Verlag, 1973, pp. 82–89.

[8] X.-C. Cai, An Additive Schwarz Algorithm for Nonselfadjoint Elliptic Equations, T. F. Chan, R. Glowinski, J. Periaux, and O. Widlund eds., *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations,* SIAM, Philadelphia, 1990, pp. 232–244.

[9] X.-C. Cai, W. D. Gropp, and D. E. Keyes, *Convergence Rate Estimate for a Domain Decomposition Method,* Technical Report 827, Yale University, Department of Computer Science, October 1990.

[10] T. F. Chan, Boundary Probe Domain Decomposition Preconditioners for Fourth Order Problems, T. F. Chan, R. Glowinski, J. Periaux, and O. Widlund eds., *Second International Symposium on Domain Decomposition Methods,* SIAM, Philadelphia, 1989, pp. 168–172.

[11] T. F. Chan and D. Goovaerts, *A Note on the Efficiency of Domain Decomposed Incomplete Factorizations,* SIAM J. Sci. Stat. Comp., 11(1990), pp. 794–803.

[12] T. F. Chan and D. E. Keyes, Interface Preconditiongs for Domain-Decomposed Convection-Diffusion Operators, T. F. Chan, R. Glowinski, J. Periaux, and O. Widlund eds., *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations,* SIAM, Philadelphia, 1990, pp. 245–262.

[13] D. Dewey and A. T. Patera, Geometry-Defining Processors for Partial Differential Equations, B. J. Alder ed., *Architectures and Performance of Specialized Computer Systems,* Academic Press, New York, 1988.

[14] M. Dryja, An Additive Schwarz Algorithm for Two- and Three-Dimensional Finite Element Elliptic Problems, T. F. Chan, R. Glowinski, J. Periaux, and O. Widlund eds., *Second International Symposium on Domain Decomposition Methods,* SIAM, Philadelphia, 1989, pp. 168–172.

[15] M. Dryja and O. B. Widlund, On the Optimality of an Additive Refinement Method, J. Mandel, S. F. McCormick, J. E. Dendy, Jr., C. Farhat, G. Lonsdale, S. V. Parter, J. W. Ruge, and K. Stuben eds., *Proceedings of the Fourth Copper Mountain Conference on Multigrid Methods,* SIAM, Philadelphia, 1989, pp. 161–170.

[16] T. Dupont, R. Kendall, and H. H. Rachford, *An Approximate Factorization Procedure for Solving Self-Adjoint Elliptic Difference Equations,* SIAM J. Num. Anal., 5 (1968), pp. 559–573.

[17] S. C. Eisenstat, H. C. Elman, M. H. Schultz, and A. H. Sherman, *The (New) Yale Sparse Matrix Package,* Technical Report 265, Yale University, Department of Computer Science, April 1983.

[18] R. E. Ewing and R. D. Lazarov, Adaptive Local Grid Refinement, *Proceedings SPE Rocky Mountain Regional Meeting,* 1988. SPE No. 17806.

[19] J. E. Flaherty, P. J. Paslow, M. S. Shepard, and J. D. Vasilakis eds., *Adaptive Methods for Partial Differential Equations,* SIAM, Philadelphia, 1989.

[20] R. W. Freund and N. M. Nachtigal, *QMR: A Quasi-Minimum Residual Method for Non-Hermitian Linear Systems,* Technical Report, RIACS, NASA Ames Research Center, 1990.

[21] A. George, *Nested Dissection of a Regular Finite Element Mesh,* SIAM J. Num. Anal., 10 (1973), pp. 345–363.

[22] W. D. Gropp, *Local Uniform Mesh Refinement for Elliptic Partial Differential Equations,* Technical Report YALE/DCS/RR-278, Yale University Dept. of Computer Science, July 1983.

[23] W. D. Gropp and D. E. Keyes, *Domain Decomposition on Parallel Computers,* Impact of Comput. in Sci. and Eng., 1 (1989), pp. 421–439.

[24] ———, *Domain Decomposition with Local Mesh Refinement,* Technical Report YALE/DCS/RR-726, Yale University Dept. of Computer Science, August 1989.

[25] ———, *Parallel Performance of Domain-Decomposed Preconditioned Krylov Methods for PDEs with Adaptive Refinement,* Technical Report YALE/DCS/RR-773, Yale University Dept. of Computer Science, April 1990.

[26] ———, *Parallel Domain Decomposition and the Solution of Nonlinear Systems of Equations,* Technical Report Mathematics and Computer Science Preprint MCS–P186–1090, Argonne National Laboratory, October 1990.

[27] ———, *Domain Decomposition Methods in Computational Fluid Dynamics,* Technical Report 91-20, ICASE, February 1991.

[28] E. N Houstis, R. E. Lynch, and J. R. Rice, *Evaluation of Numerical Methods for Elliptic Partial Differential Equations,* J. Comp. Phys., 27 (1978), pp. 323–350.

[29] H. Jarausch, *On An Adaptive Grid Refining Technique for Finite Element Approximations,* SIAM J. Sci. Stat. Comp., 7 (1986), pp. 1105–1120.

[30] D. E. Keyes and W. D. Gropp, *A Comparison of Domain Decomposition Techniques for Elliptic Partial Differential Equations and Their Parallel Implementation,* SIAM J. Sci. Stat. Comp., 8 (1987), pp. s166-s202.

[31] ———, Domain Decomposition Techniques for Nonsymmetric Systems of Elliptic Boundary Value Problems: Examples from CFD, T. F. Chan, R. Glowinski, J. Periaux, and O. Widlund eds., *Second International Symposium on Domain Decomposition Methods,* SIAM, Philadelphia, 1989, pp. 321–339.

[32] ———, *Domain Decomposition Techniques for the Parallel Solution of Nonsymmetric Systems of Elliptic BVPs,* Appl. Num. Math., 6 (1990), pp. 281–301.

[33] ————, *Domain-Decomposable Preconditioners for Second-Order Upwind Discretizations of Multicomponent Systems,* Technical Report Mathematics and Computer Science Preprint MCS–P187–1090, Argonne National Laboratory, October 1990.

[34] Y. Maday, C. Mavriplis, and A. T. Patera, Nonconforming Mortar Element Methods: Application to Spectral Discretizations, T. F. Chan, R. Glowinski, J. Periaux, and O. Widlund eds., *Second International Symposium on Domain Decomposition Methods,* SIAM, Philadelphia, 1989, pp. 392–418.

[35] T. A. Manteuffel and S. V. Parter, *Preconditioning and Boundary Conditions,* Technical Report LA-UR-88-2626, Los Alamos National Laboratory, July 1988.

[36] S. F. McCormick, *Multilevel Adaptive Methods For Partial Differential Equations,* SIAM, Philadelphia, 1983.

[37] S. McCormick and D. Quinlan, *Asynchronous Multilevel Adaptive Methods for Solving Partial Differential Equations on Multiprocessors: Performance Results,* Par. Comput., 12(1989), pp. 145–156.

[38] J. A. Meijerink and H. A. Van der Vorst, *Guidelines for the Usage of Incomplete Decompositions in Solving Sets of Linear Equations As They Occur in Practical Problems,* J. Comp. Phys., 44(1981), pp. 134–155.

[39] W. Proskurowski, Remarks on the Spectral Equivalence of Certain Discrete Operators, T. F. Chan, R. Glowinski, J. Periaux, and O. Widlund eds., *Second International Symposium on Domain Decomposition Methods,* SIAM, Philadelphia, 1989, pp. 103–113.

[40] J. R. Rice, E. N. Houstis, and W. R. Dyksen, *A Population of Linear Second Order, Elliptic Partial Differential Equations on Rectangular Domains - Part I,* Technical Report 2078, Mathematics Research Center, Univ. of Wisconsin - Madison, May 1980.

[41] ————, *A Population of Linear Second Order, Elliptic Partial Differential Equations on Rectangular Domains - Part II,* Technical Report 2079, Mathematics Research Center, Univ. of Wisconsin - Madison, May 1980.

[42] Y. Saad and M. H. Schultz, *GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems,* SIAM J. Sci. Stat. Comp., 7(1986), pp. 856–869.

[43] B. F. Smith, *Domain Decomposition Algorithms for the Partial Differential Equations of Linear Elasticity,* Technical Report 517, Courant Institute, NYU, September 1990.

[44] B. Swartz, *Courant-Like Conditions Limit Reasonable Mesh Refinement to Order h^2,* SIAM J. Sci. Stat. Comp., 8(1987), pp. 924–933.

[45] H. A. Van der Vorst, *Bi-CCSTAB: A More Smoothly Converging Variant of CG-S for the Solution of Nonsymmetric Linear Systems,* 1990. (Preprint).

[46] J. W. Watts, III, *A Conjugate Gradient-Truncated Direct Method for the Iterative Solution of the Reservoir Simulation Pressure Equation,* Soc. Petrol. Engin. J., 21(1981), pp. 345–353.

[47] H. Yserentant, *On the Multi-level Splitting of Finite Element Spaces for Indefinite Elliptic Boundary Value Problems,* SIAM J. Num. Anal., 23(1986), pp. 581–595.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-187528 ICASE Report No. 91-19		

4. Title and Subtitle	5. Report Date
DOMAIN DECOMPOSITION WITH LOCAL MESH REFINEMENT	February 1991
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
William D. Gropp David E. Keyes	91-19
	10. Work Unit No.
	505-90-52-01

9. Performing Organization Name and Address	11. Contract or Grant No.
Institute for Computer Applications in Science and Engineering Mail Stop 132C, NASA Langley Research Center Hampton, VA 23665-5225	NAS1-18605
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Contractor Report
National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665-5225	14. Sponsoring Agency Code

15. Supplementary Notes

Langley Technical Monitor:
Michael F. Card

Submitted to SIAM Journal of Scientific and Statistical Computation

Final Report

16. Abstract

We describe a preconditioned Krylov iterative algorithm based on domain decomposition for linear systems arising from implicit finite-difference or finite-element discretizations of partial differential equation problems requiring local mesh refinement. To keep data structures as simple as possible for parallel computing applications, we define the fundamental computational unit in the algorithm as a subregion of the domain spanned by a locally uniform tensor-product grid, called a tile. In the tile-based domain decomposition approach, two levels of discretization are considered at each point of the domain: a global coarse grid defined by tile vertices only, and a local fine grid where the degree of resolution can vary from tile to tile. One global level and one local level provide the flexibility required to adaptively discretize a diverse collection of problems on irregular regions and solve them at convergence rates that deteriorate only logarithmically in the finest mesh parameter, with the coarse tessellation held fixed. A logarithmic departure from optimality seems to be a reasonable compromise for the simplicity of the composite grid data structure and concomitant regular data exchange patterns in a multiprocessor environment. We report some experiments with up to 1024 tiles, comment on the evolution of the algorithm, and contrast it with optimal nonrefining two-level algorithms and optimal refining multilevel algorithms. Computational comparisons with some other popular methods are presented.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
domain decomposition, preconditioning, Krylov methods, mesh refinement, elliptic problems	64 - Numerical Analysis Unclassified - Unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	32	A03

NASA FORM 1626 OCT 86